

XML canónico versión 1.0

Recomendación del W3C del 15 de marzo de 2001

Esta versión:
http://www.w3.org/TR/2001/REC-xml-c14n-20010315
Ultima versión:
http://www.w3.org/TR/xml-c14n
Versión previa:
http://www.w3.org/TR/2001/PR-xml-c14n-20010119
Autor/Editor:
John Boyer, PureEdge Solutions Inc., jboyer@PureEdge.com

Abstracto

Cualquier documento XML es parte de un conjunto de documentos XML que son lógicamente equivalentes dentro del contexto de una aplicación, pero que varían en su representación física según los cambios sintácticos permitidos por XML 1.0 [XML] y los espacios de nombres en XML [Nombres] . Esta especificación describe un método para generar una representación física, la forma canónica, de un documento XML que tiene en cuenta los cambios permitidos. Salvo limitaciones relativas a algunos casos inusuales, si dos documentos tienen la misma forma canónica, entonces los dos documentos son lógicamente equivalentes dentro del contexto de aplicación dado. Tenga en cuenta que dos documentos pueden tener diferentes formas canónicas y aun así ser equivalentes en un contexto determinado según reglas de equivalencia específicas de la aplicación que ninguna especificación XML generalizada podría tener en cuenta.

Estado de este documento

Esta sección describe el estado de este documento en el momento de su publicación. Otros documentos pueden reemplazar este documento. El estado más reciente de esta serie de documentos se mantiene en el W3C.

Este documento ha sido revisado por miembros del W3C y otras partes interesadas y ha sido respaldado por el Director como Recomendación del W3C . Es un documento estable y puede usarse como material de referencia o citarse como referencia normativa de otro documento.

Este documento ha sido elaborado por el Grupo de trabajo sobre firmas XML del IETF/W3C (consulte también la Declaración de actividad de firmas XML del W3C). Esta

versión incluye algunas mejoras editoriales menores con respecto a la versión anterior. El único cambio sustancial es la adición de una referencia al corrigendum [NFC-Corrigendum] de *TR15, Formularios de normalización Unicode* [NFC]. Este corrigendum corrige un error por el cual el carácter U+FB1D LETRA HEBREA YOD CON HIRIQ se omitió por error de las Exclusiones de composición de *Unicode 3.0* . Las implementaciones XML canónicas ahora deben excluir (correctamente) este carácter de la composición de caracteres durante el procesamiento [NFC].

La especificación Canonical XML fue revisada exhaustivamente durante su desarrollo, según lo dispuesto por el Proceso W3C. El Grupo de Trabajo resolvió con éxito todos los problemas planteados durante la última convocatoria y la convocatoria de implementación y documentó la existencia de implementaciones interoperables en su informe de interoperabilidad .

Informe los errores en este documento al editor y envíe una copia a: la lista de correo electrónico pública w3c-ietf-xmldsig@w3.org . Cualquier error de este tipo se documentará en una errata disponible en http://www.w3.org/2001/03/C14N-errata .

Puede encontrar una lista de todos los informes técnicos actuales del W3C en http://www.w3.org/TR .

Tabla de contenido

---

1. Introducción

La Recomendación XML 1.0 [XML] especifica la sintaxis de una clase de recursos llamados documentos XML. La Recomendación sobre espacios de nombres en XML [Nombres] especifica sintaxis y semántica adicionales para documentos XML. Es posible que los documentos XML que son equivalentes para los fines de muchas aplicaciones difieran en su representación física. Por ejemplo, pueden diferir en la estructura de su entidad, el orden de los atributos y la codificación de caracteres. El objetivo de esta especificación es establecer un método para determinar si dos documentos son idénticos o si una aplicación no ha cambiado un documento, excepto las transformaciones permitidas por XML 1.0 y los espacios de nombres en XML.

1.1 Terminología

Las palabras clave "DEBE", "NO DEBE", "REQUERIDO", "DEBE", "NO DEBE", "DEBE", "NO DEBE", "RECOMENDADO", "PUEDE" y "OPCIONAL" en este documento son debe interpretarse como se describe en RFC 2119 [Palabras clave] .

Consulte [Nombres] para conocer la definición de QName .

Un *subconjunto de documentos* es una parte de un documento XML indicado por un conjunto de nodos que puede no incluir todos los nodos del documento.

La *forma canónica* de un documento XML es la representación física del documento producido mediante el método descrito en esta especificación. Los cambios se resumen en la siguiente lista:

- El documento está codificado en UTF-8.
- Saltos de línea normalizados a #xA en la entrada, antes del análisis
- Los valores de los atributos están normalizados, como si lo hiciera un procesador de validación.
- Se reemplazan las referencias de caracteres y entidades analizadas.
- Las secciones CDATA se reemplazan con su contenido de caracteres.
- Se eliminan la declaración XML y la declaración de tipo de documento (DTD).
- Los elementos vacíos se convierten en pares de etiquetas de inicio y fin.
- Los espacios en blanco fuera del elemento del documento y dentro de las etiquetas de inicio y fin están normalizados.
- Se conservan todos los espacios en blanco en el contenido de los caracteres (excluidos los caracteres eliminados durante la normalización del avance de línea)
- Los delimitadores de valores de atributos se establecen entre comillas (comillas dobles)
- Los caracteres especiales en los valores de los atributos y el contenido de los caracteres se reemplazan por referencias de caracteres.
- Se eliminan las declaraciones de espacios de nombres superfluas de cada elemento.
- Los atributos predeterminados se agregan a cada elemento.
- El orden lexicográfico se impone en las declaraciones de espacios de nombres y atributos de cada elemento.

El término *XML canónico* se refiere a XML que está en formato canónico. El *método de canonicalización XML* es el algoritmo definido por esta especificación que genera la forma canónica de un documento XML o subconjunto de documentos determinado. El término *canonicalización XML* se refiere al proceso de aplicar el método de canonicalización XML a un documento o subconjunto de documentos XML.

La Recomendación XPath 1.0 [XPath] define el término *conjunto de nodos* y especifica un modelo de datos para representar un documento XML de entrada como un conjunto de nodos de varios tipos (elemento, atributo, espacio de nombres, texto, comentario, instrucción de procesamiento y raíz). Los nodos se incluyen o excluyen de un conjunto de nodos según la evaluación de una expresión. Dentro de esta especificación, un conjunto de nodos se utiliza para indicar directamente si cada nodo debe representarse o no en la forma canónica (en este sentido, se utiliza como un conjunto matemático formal). Un nodo excluido del conjunto no se representa en la forma canónica que se genera, incluso si su nodo principal está incluido en el conjunto de nodos. Sin embargo, un nodo omitido aún puede afectar la representación de sus descendientes (por ejemplo, aumentando el contexto del espacio de nombres de los descendientes).

1.2 Aplicaciones

Dado que la Recomendación XML 1.0 [XML] y los espacios de nombres en la Recomendación XML [Nombres] definen múltiples métodos sintácticos para expresar la misma información, las aplicaciones XML tienden a tomarse libertades con cambios que no tienen impacto en el contenido de información del documento. La canonicalización XML está diseñada para ser útil en aplicaciones que requieren la capacidad de probar si se ha modificado el contenido de información de un documento o subconjunto de documentos. Esto se hace comparando la forma canónica del documento original antes del procesamiento de la solicitud con la forma canónica del documento resultante del procesamiento de la solicitud.

Por ejemplo, una firma digital sobre la forma canónica de un documento XML o un subconjunto de documentos permitiría que los cálculos del resumen de firmas no tuvieran en cuenta los cambios en la representación física del documento original, siempre que los cambios se definan como lógicamente equivalentes en XML 1.0 o Espacios de nombres en XML. Durante la generación de la firma, el resumen se calcula sobre la forma canónica del documento. A continuación, el documento se transfiere a la parte que confía, que valida la firma leyendo el documento y calculando un resumen de la forma canónica del documento recibido. La equivalencia de los resúmenes calculados por las partes firmantes y confiantes (y, por tanto, la equivalencia de las formas canónicas sobre las cuales fueron calculados) garantiza que el contenido informativo del documento no haya sido alterado desde su firma.

1.3 Limitaciones

Dos documentos XML pueden tener contenido de información diferente que, sin embargo, es lógicamente equivalente dentro de un contexto de aplicación determinado. Aunque dos documentos XML son equivalentes (aparte de las limitaciones dadas en esta sección) si sus formas canónicas son idénticas, no es un objetivo de este trabajo establecer un método tal que dos documentos XML sean equivalentes si y sólo si sus formas canónicas son *idénticas* . . Este método es inalcanzable, en parte debido a reglas específicas de la aplicación, como las que rigen los espacios en blanco sin importancia y datos equivalentes (por ejemplo, `<color>black</color>`versus `<color>rgb(0,0,0) </color>`). También existen equivalencias establecidas por otras Recomendaciones y Borradores de Trabajo del W3C. La contabilidad de estas reglas de equivalencia adicionales está fuera del alcance de este trabajo. Pueden ser aplicados por la aplicación o convertirse en objeto de futuras especificaciones.

Es posible que la forma canónica de un documento XML no sea completamente operativa dentro del contexto de la aplicación, aunque las circunstancias bajo las cuales esto ocurre son inusuales. Este problema puede ser motivo de preocupación en

determinadas aplicaciones, ya que la forma canónica de un documento y la forma canónica del documento son equivalentes. Por ejemplo, en una solicitud de firma digital, no se puede establecer si se firmó el documento original operativo o la forma canónica no operativa porque la forma canónica se puede sustituir por el documento original sin cambiar el cálculo del resumen. Sin embargo, el riesgo de seguridad sólo ocurre en las circunstancias inusuales que se describen a continuación, las cuales pueden resolverse o al menos detectarse antes de la generación de la firma digital.

Las dificultades surgen debido a la pérdida de la siguiente información no disponible en el modelo de datos :

1. URI base, especialmente en contenido derivado del texto de reemplazo de referencias de entidades analizadas generales externas
2. notaciones y referencias de entidades externas no analizadas
3. tipos de atributos en la declaración de tipo de documento

En el primer caso, tenga en cuenta que un documento que contiene un URI relativo [URI] solo es operativo cuando se accede desde un URI específico que proporciona el URI base adecuado. Además, si el documento contiene referencias de entidades externas generales analizadas a contenido que contiene URI relativos, entonces los URI relativos no estarán operativos en la forma canónica, que reemplaza la referencia de entidad con contenido interno (cambiando así implícitamente el URI base predeterminado de ese contenido).). Ambos problemas generalmente se pueden resolver agregando soporte para el `xml:base`atributo [XBase] a la aplicación y luego agregando `xml:base`atributos apropiados al elemento del documento y a todos los elementos de nivel superior en entidades externas. Además, las aplicaciones suelen tener la oportunidad de resolver URI relativos antes de que sea necesario un formato canónico. Por ejemplo, en una aplicación de firma digital, un documento suele recuperarse y procesarse antes de generar la firma. El procesamiento DEBE crear un nuevo documento en el que los URI relativos se hayan convertido en URI absolutos, mitigando así cualquier riesgo de seguridad para el nuevo documento.

En el segundo caso, la pérdida de referencias de entidades externas no analizadas y las notaciones que las vinculan a las aplicaciones significa que las formas canónicas no pueden distinguir adecuadamente entre documentos XML que incorporan datos no analizados a través de este mecanismo. Este es un caso inusual precisamente porque la mayoría de los procesadores XML actualmente descartan la declaración de tipo de documento, lo que descarta la notación, la vinculación de la entidad a un URI y el tipo de atributo que vincula el valor del atributo a un nombre de entidad. Para los documentos que deben someterse a más de un procesador XML, el diseño XML normalmente indica una referencia a datos no analizados utilizando un URI en el valor del atributo.

En el tercer caso, la pérdida de tipos de atributos puede afectar la forma canónica de diferentes maneras según el tipo. Los atributos de tipo ID dejan de ser atributos de ID. Por lo tanto, cualquier expresión XPath que haga referencia a la forma canónica utilizando la `id()`función deja de funcionar. Los tipos de atributos ENTIDAD y ENTIDADES no forman parte de este caso; están cubiertos en el segundo caso anterior. Los atributos de tipo enumerado y de tipo ID, IDREF, IDREFS, NMTOKEN, NMTOKENS y NOTATION no se restringen adecuadamente durante futuros intentos de cambiar el valor del atributo si la forma canónica reemplaza el documento original durante el procesamiento de la solicitud. Las aplicaciones pueden evitar las dificultades de este caso asegurándose de anteponer una declaración de tipo de documento adecuada antes de utilizar la forma canónica en un procesamiento XML posterior. Es probable que esto sea una tarea fácil ya que las listas de atributos generalmente se adquieren a partir de un subconjunto de DTD externo estándar, y cualquier declaración de entidad y notación que

no esté también en el subconjunto de DTD externo generalmente se construye a partir de información de configuración de la aplicación y se agrega al subconjunto de DTD interno.

Si bien estas limitaciones no son graves, sería posible resolverlas en una versión futura de canonicalización XML si, por ejemplo, se creara una nueva versión de XPath basada en el conjunto de información XML [Infoset] actualmente en desarrollo en el W3C.

2 Canonicalización XML

2.1 Modelo de datos

El modelo de datos definido en la Recomendación XPath 1.0 [XPath] se utiliza para representar el documento XML de entrada o el subconjunto de documentos. Las implementaciones DEBEN, pero no necesariamente, basarse en una implementación XPath. La canonicalización XML se define en términos de la definición XPath de un conjunto de nodos, y las implementaciones DEBEN producir resultados equivalentes.

El primer parámetro de entrada al método de canonicalización XML es un conjunto de nodos XPath o una secuencia de octetos que contiene un documento XML bien formado. Las implementaciones DEBEN soportar la entrada de flujo de octetos y DEBEN soportar también la característica de subconjunto de documentos a través de la entrada de conjuntos de nodos. Con el fin de describir la canonicalización en términos de un conjunto de nodos XPath, esta sección describe cómo se convierte un flujo de octetos en un conjunto de nodos XPath.

El segundo parámetro de entrada al método de canonicalización XML es un indicador booleano que indica si los comentarios deben incluirse o no en la salida del formulario canónico mediante el método de canonicalización XML. Si una forma canónica contiene comentarios correspondientes a los nodos de comentarios en el conjunto de nodos de entrada, el resultado se denomina *XML canónico con comentarios* . Tenga en cuenta que el modelo de datos XPath no crea nodos de comentarios para los comentarios que aparecen dentro de la declaración de tipo de documento (DTD). Se REQUIEREN implementaciones que sean capaces de producir XML canónico excluyendo todos los comentarios que puedan haber aparecido en el documento de entrada o subconjunto de documentos. Se RECOMIENDA la compatibilidad con XML canónico con comentarios.

Si un documento XML debe convertirse en un conjunto de nodos, XPath REQUIERE que se utilice un procesador XML para crear los nodos de su modelo de datos para representar completamente el documento. El procesador XML realiza las siguientes tareas en orden:

1. normalizar los avances de línea
2. normalizar valores de atributos
3. reemplazar las secciones CDATA con su contenido de caracteres
4. resolver referencias de caracteres y entidades analizadas

El flujo de octetos de entrada DEBE contener un documento XML bien formado, pero no es necesario validar la entrada. Sin embargo, la normalización del valor del atributo y la resolución de la referencia de la entidad DEBEN realizarse de acuerdo con los comportamientos de un procesador XML de validación. Además, en cada elemento se crean nodos para atributos predeterminados (declarados en ATTLIST con un AttValue pero no especificados). Por lo tanto, las declaraciones en la declaración de tipo de documento se utilizan para ayudar a crear la forma canónica, aunque la declaración de tipo de documento no se conserve en la forma canónica.

El modelo de datos XPath representa datos utilizando caracteres UCS. Las implementaciones DEBEN utilizar procesadores XML que admitan UTF-8 y UTF-16 y traducir al dominio de caracteres UCS. Para UTF-16, la marca de orden de bytes inicial se trata como un artefacto de codificación y se elimina de los datos de caracteres UCS (los espacios sin separación de ancho cero posteriores que aparecen dentro de los datos UTF-16 no se eliminan) [UTF-16, Sección 3.2] . Se RECOMIENDA la compatibilidad con la codificación ISO-8859-1 y todas las demás codificaciones de caracteres son OPCIONALES.

Todos los espacios en blanco dentro del elemento del documento raíz DEBEN conservarse (excepto los caracteres #xD eliminados por la normalización del delimitador de línea). Esto incluye todos los espacios en blanco en entidades externas. Los espacios en blanco fuera del elemento del documento raíz DEBEN descartarse.

En el modelo de datos XPath existen los siguientes tipos de nodos: raíz, elemento, comentario, instrucción de procesamiento, texto, atributo y espacio de nombres. Existe un único nodo raíz cuyos hijos procesan nodos de instrucciones y nodos de comentarios para representar información fuera del elemento del documento (y fuera de la declaración del tipo de documento). El nodo raíz también tiene un nodo de elemento único que representa el elemento del documento de nivel superior. Cada nodo de elemento puede tener nodos secundarios de tipo elemento, texto, instrucción de procesamiento y comentario. Los atributos y espacios de nombres asociados con un elemento no se consideran nodos secundarios del elemento, pero se asocian con el elemento mediante su inclusión en los ejes de atributos y espacios de nombres del elemento. Tenga en cuenta que los ejes de atributos y espacios de nombres pueden no corresponderse directamente con el texto que aparece en la etiqueta de inicio del elemento en el documento original.

Nota: Un elemento tiene nodos de atributos para representar las declaraciones de atributos que no pertenecen al espacio de nombres que aparecen en su etiqueta de inicio, *así como* nodos para representar los atributos predeterminados.

En virtud del modelo de datos XPath, la canonicalización XML tiene en cuenta el espacio de nombres [Nombres] . Sin embargo, no puede y, por lo tanto, no tiene en cuenta las equivalencias de espacios de nombres mediante la reescritura de prefijos de espacios de nombres (consulte la explicación en la Sección 4). En el modelo de datos XPath, cada elemento y atributo tiene un nombre devuelto por la función `name()`que puede, a discreción de la aplicación, ser el QName que aparece en el documento original. La canonicalización XML REQUIERE que el procesador XML conserve suficiente información para que se pueda proporcionar el QName del elemento tal como apareció en el documento original.

Nota: Un elemento ***E*** tiene nodos de espacio de nombres que representan sus declaraciones de espacio de nombres, *así como* cualquier declaración de espacio de nombres realizada por sus antepasados que no hayan sido anuladas en las declaraciones de ***E*** , el espacio de nombres predeterminado si no está vacío y la declaración del prefijo. `xml`.

Nota: Esta especificación respalda la reciente decisión plenaria de XML de desaprobar los URI de espacio de nombres relativos de la siguiente manera: las implementaciones de canonicalización XML DEBEN informar una falla de operación en documentos que contienen URI de espacio de nombres relativos. La canonicalización XML NO DEBE implementarse con un analizador XML que convierta URI relativos en URI absolutos.

El contenido de los caracteres se representa en el modelo de datos XPath con nodos de texto. Todos los caracteres consecutivos se colocan en un único nodo de texto. Además,

los caracteres del nodo de texto están representados en el dominio de caracteres UCS. El método de canonicalización XML no realiza la normalización del modelo de caracteres (consulte la explicación en la Sección 4). Sin embargo, el procesador XML utilizado para preparar la entrada del modelo de datos XPath DEBE utilizar el formulario de normalización Unicode C [NFC , NFC-Corrigendum] al convertir un documento XML al dominio de caracteres UCS desde cualquier codificación que no esté basada en UCS (actualmente, Las codificaciones basadas en UCS incluyen UTF-8, UTF-16, UTF-16BE y UTF-16LE, UCS-2 y UCS-4).

Dado que la canonicalización XML convierte un conjunto de nodos XPath en una forma canónica, el primer parámetro DEBE ser un conjunto de nodos XPath o debe convertirse de una secuencia de octetos a un conjunto de nodos realizando el procesamiento XML necesario para crear los nodos XPath. descrito anteriormente, luego estableciendo un contexto de evaluación XPath inicial de:

- Un **nodo de contexto** , inicializado en el nodo raíz del documento XML de entrada.
- Una **posición de contexto** , inicializada en 1.
- Un **tamaño de contexto** , inicializado en 1.
- Cualquier **biblioteca de funciones** que cumpla con la Recomendación XPath.
- Un conjunto vacío de **enlaces de variables** .
- Un conjunto vacío de **declaraciones de espacios de nombres** .

y evaluando la siguiente expresión predeterminada:

Valor del parámetro comentario	**Expresión XPath predeterminada**
Sin (falso)	`(//. | //@* | //namespace::*)[not(self::comment())]`
Con (verdadero)	`(//. | //@* | //namespace::*)`

Las expresiones de esta tabla generan un conjunto de nodos que contiene cada nodo del documento XML (excepto los comentarios si el valor del parámetro comentario es falso).

Si la entrada es un conjunto de nodos XPath, entonces el conjunto de nodos debe contener explícitamente cada nodo que se representará en la forma canónica. Por ejemplo, el resultado de la expresión XPath `id("E")`es un conjunto de nodos que contiene sólo el nodo correspondiente al elemento con un valor de atributo ID de "E". Dado que ninguno de sus nodos descendientes, nodos de atributos y nodos de espacio de nombres están en el conjunto, la forma canónica consistiría únicamente en las etiquetas inicial y final del elemento, menos las declaraciones de atributo y espacio de nombres, sin contenido interno. La Sección 3.7 ejemplifica cómo serializar un elemento identificado junto con su contenido interno, atributos y declaraciones de espacio de nombres.

2.2 Orden de los documentos

Aunque un conjunto de nodos XPath se define como desordenado, la Recomendación XPath 1.0 [XPath] define el término *orden del documento* como el orden en el que aparece el primer carácter de la representación XML de cada nodo en la representación XML del documento después de la expansión. de entidades generales, excepto para los nodos de espacio de nombres y atributos cuyo orden de documentos depende de la aplicación.

El método de canonicalización XML procesa un conjunto de nodos imponiendo las siguientes reglas adicionales de orden de documentos en los nodos de espacio de nombres y atributos de cada elemento:

- Los nodos de espacio de nombres y atributos de un elemento tienen una posición en el orden del documento mayor que la del elemento pero menor que cualquier nodo secundario del elemento.
- Los nodos de espacio de nombres tienen una posición de orden de documentos menor que los nodos de atributos.
- Los nodos de espacio de nombres de un elemento se ordenan lexicográficamente por nombre local (el nodo de espacio de nombres predeterminado, si existe, no tiene nombre local y, por lo tanto, es lexicográficamente el menos).
- Los nodos de atributos de un elemento se ordenan lexicográficamente con el URI del espacio de nombres como clave principal y el nombre local como clave secundaria (un URI de espacio de nombres vacío es el menos lexicográficamente).

La comparación lexicográfica, que ordena las cadenas alfabéticamente de menor a mayor, se basa en los valores de puntos de código UCS, lo que equivale al orden lexicográfico basado en UTF-8.

2.3 Modelo de procesamiento

El conjunto de nodos XPath se convierte en un flujo de octetos, la forma canónica, generando los caracteres UCS representativos para cada nodo en el conjunto de nodos en orden ascendente del documento y luego codificando el resultado en UTF-8 (sin una marca de orden de bytes inicial).). Ningún nodo se procesa más de una vez. Tenga en cuenta que el procesamiento de un nodo de elemento ***E*** incluye el procesamiento de todos los miembros del conjunto de nodos del cual ***E*** es un ancestro. Por lo tanto, inmediatamente después de generar el texto representativo de ***E*** **,** ***E*** y todos los nodos de los cuales ***E*** es un ancestro se eliminan del conjunto de nodos (o se produce alguna operación lógicamente equivalente de modo que el siguiente nodo del conjunto de nodos en el orden del documento no haya sido eliminado). procesada). Sin embargo, tenga en cuenta que un nodo de elemento no se elimina del conjunto de nodos hasta que se procesan sus elementos secundarios.

El resultado del procesamiento de un nodo depende de su tipo y de si está o no en el conjunto de nodos. Si un nodo no está en el conjunto de nodos, entonces no se genera ningún texto para el nodo excepto el resultado del procesamiento de su espacio de nombres y ejes de atributos (solo elementos) y sus hijos (elementos y el nodo raíz). Si el nodo está en el conjunto de nodos, entonces se genera texto para representar el nodo en la forma canónica además del texto generado al procesar el espacio de nombres y los ejes de atributos y los nodos secundarios del nodo.

NOTA: El conjunto de nodos se trata como un conjunto de nodos, no como una lista de subárboles. Para canonicalizar un elemento, incluidos sus espacios de nombres, atributos y contenido, el conjunto de nodos debe contener todos los nodos correspondientes a estas partes del documento, no solo el nodo del elemento.

El texto generado para un nodo depende del tipo de nodo y se muestra en la siguiente lista:

- **Nodo raíz:** el nodo raíz es el padre del elemento de documento de nivel superior. El resultado del procesamiento de cada uno de sus nodos secundarios que se encuentra en el conjunto de nodos en el orden del documento. El nodo raíz no

genera una marca de orden de bytes, una declaración XML ni nada dentro de la declaración de tipo de documento.

- **Nodos de elementos:** si el elemento no está en el conjunto de nodos, entonces el resultado se obtiene procesando el eje del espacio de nombres, luego el eje de atributos y luego procesando los nodos secundarios del elemento que están en el conjunto de nodos (en el orden del documento). . Si el elemento está en el conjunto de nodos, entonces el resultado es un corchete angular abierto (<), el elemento QName, el resultado del procesamiento del eje del espacio de nombres, el resultado del procesamiento del eje de atributos, un corchete angular cerrado (>), el resultado de procesar los nodos secundarios del elemento que están en el conjunto de nodos (en el orden del documento), un corchete angular abierto, una barra diagonal (/), el elemento QName y un corchete angular cerrado.
    - *Eje del espacio de nombres:* considere una lista ***L*** que contenga solo nodos de espacio de nombres en el eje y en el conjunto de nodos en orden lexicográfico (ascendente). Para comenzar a procesar ***L*** , si el primer nodo no es el nodo del espacio de nombres predeterminado (un nodo sin URI de espacio de nombres ni nombre local), genere un espacio seguido de `xmlns=""` *si y solo* si se cumplen las siguientes condiciones:

        - el elemento ***E*** que posee el eje está en el conjunto de nodos
        - El elemento ancestro más cercano de ***E*** en el conjunto de nodos tiene un nodo de espacio de nombres predeterminado en el conjunto de nodos (los nodos de espacio de nombres predeterminados siempre tienen valores no vacíos en XPath)

        La última condición elimina las apariciones innecesarias de `xmlns=""`en la forma canónica, ya que un elemento solo recibe un `xmlns=""`si su espacio de nombres predeterminado está vacío y si tiene un padre inmediato en la forma canónica que tiene un espacio de nombres predeterminado no vacío. Para finalizar el procesamiento ***L*** , simplemente procese cada nodo del espacio de nombres en ***L*** , excepto omitir el nodo del espacio de nombres con el nombre local `xml`, que define el `xml`prefijo, si su valor de cadena es `http://www.w3.org/XML/1998/namespace`.

    - *Eje de atributos:* en orden lexicográfico (ascendente), procese cada nodo que se encuentra en el eje de atributos del elemento y en el conjunto de nodos.
- **Nodos de espacio de nombres:** un nodo de espacio de nombres ***N*** se ignora si el elemento ancestro más cercano del elemento principal del nodo que está en el conjunto de nodos tiene un nodo de espacio de nombres en el conjunto de nodos con el mismo nombre local y valor que ***N.*** De lo contrario, procese el nodo de espacio de nombres ***N*** de la misma manera que un nodo de atributo, excepto que asigne el nombre local `xmlns`al nodo de espacio de nombres predeterminado si existe (en XPath, el nodo de espacio de nombres predeterminado tiene un URI y un nombre local vacíos).
- **Nodos de atributos:** un espacio, el QName del nodo, un signo igual, una comilla abierta (comilla doble), el valor de cadena modificado y una comilla cerrada (comilla doble). El valor de cadena del nodo se modifica reemplazando todos los símbolos (&) por `&`, todos los corchetes angulares abiertos (<) por `<`, todos los caracteres de comillas por `"`y los espacios en blanco #x9, #xA y #xD con referencias de caracteres. Las referencias de caracteres se escriben en mayúsculas hexadecimales sin ceros a la izquierda (por ejemplo, #xD está representado por la referencia de caracteres `
`).
- **Nodos de texto** : el valor de la cadena, excepto que todos los símbolos se reemplazan por `&`, todos los corchetes angulares abiertos (<) se reemplazan

por <, todos los corchetes angulares de cierre (>) se reemplazan por >y todos los caracteres #xD se reemplazan por
.

- **Nodos de instrucciones de procesamiento (PI):** el símbolo PI de apertura (<?), el nombre de destino de PI del nodo, un espacio inicial y el valor de la cadena si no está vacío, y el símbolo PI de cierre (?>). Si el valor de la cadena está vacío, no se agrega el espacio inicial. Además, se representa un #xA final después del símbolo PI de cierre para los PI secundarios del nodo raíz con un orden de documento menor que el elemento del documento, y un #xA inicial se representa antes del símbolo PI de apertura de los PI secundarios del nodo raíz con un orden de documento mayor que el elemento del documento.
- **Nodos de comentarios:** nada si se genera XML canónico sin comentarios. Para XML canónico con comentarios, genere el símbolo de comentario de apertura (<!--), el valor de cadena del nodo y el símbolo de comentario de cierre (-->). Además, se representa un #xA final después del símbolo de comentario de cierre para los comentarios secundarios del nodo raíz con un orden de documento menor que el elemento del documento, y un #xA inicial se representa antes del símbolo de comentario de apertura de los comentarios secundarios del nodo raíz con un orden de documento mayor que el elemento del documento. (Los comentarios secundarios del nodo raíz representan comentarios fuera del elemento de documento de nivel superior y fuera de la declaración de tipo de documento).

El QName de un nodo es el nombre local si la cadena de prefijo del espacio de nombres está vacía o el prefijo del espacio de nombres, dos puntos y luego el nombre local del elemento. El prefijo de espacio de nombres utilizado en QName DEBE ser el mismo que apareció en el documento de entrada.

2.4 Subconjuntos de documentos

Algunas aplicaciones requieren la capacidad de crear una representación física para un subconjunto de documentos XML (distinto del generado de forma predeterminada, que puede ser un subconjunto adecuado del documento si se omiten los comentarios). Las implementaciones de canonicalización XML basadas en XPath pueden proporcionar esta funcionalidad con poca sobrecarga adicional al aceptar un conjunto de nodos como entrada en lugar de un flujo de octetos.

El procesamiento de un nodo de elemento ***E*** DEBE modificarse ligeramente cuando se proporciona un conjunto de nodos XPath como entrada y el padre del elemento se omite del conjunto de nodos. Se mejora el método para procesar el eje de atributos de un elemento ***E*** en el conjunto de nodos. Todos los nodos de elementos a lo largo del eje ***E***ancestor se examinan para detectar las apariciones más cercanas de atributos en el xmlespacio de nombres, como xml:langy xml:space(estén o no en el conjunto de nodos). De esta lista de atributos, elimine los que estén en el eje de atributos de ***E*** (estén o no en el conjunto de nodos). Luego, combine lexicográficamente esta lista de atributos con los nodos del eje de atributos de ***E*** que están en el conjunto de nodos. El resultado de visitar el eje de atributos se calcula procesando los nodos de atributos en esta lista de atributos fusionados.

NOTA: Las entidades XML pueden derivar significados específicos de la aplicación desde cualquier lugar del marcado XML, así como mediante reglas no expresadas en XML 1.0 y los espacios de nombres en las Recomendaciones XML. Claramente, estas reglas no se pueden especificar en este documento, por lo que el creador del conjunto de nodos de entrada debe ser responsable de preservar la información necesaria para capturar la semántica completa de los miembros del conjunto de nodos resultante.

El XML canónico generado para un documento XML completo está bien formado. Es posible que la forma canónica de un subconjunto de documentos XML no sea XML bien formado. Sin embargo, dado que la forma canónica puede estar sujeta a un procesamiento XML adicional, la mayoría de los conjuntos de nodos XPath proporcionados para la canonicalización se diseñarán para producir una forma canónica que sea un documento XML bien formado o una entidad externa analizada general. Ya sea de un documento completo o de un subconjunto de documentos, si la forma canónica es XML bien formado, las aplicaciones posteriores del mismo método de canonicalización XML a la forma canónica no realizan cambios.

3 ejemplos de canonicalización XML

Los ejemplos de esta sección suponen un procesador sin validación, principalmente para que una declaración de tipo de documento pueda usarse para declarar entidades, así como atributos predeterminados y atributos de varios tipos (como ID y enumerados) sin tener que declarar todos los atributos para todos. elementos en el documento. Además, un ejemplo contiene un elemento que viola deliberadamente una restricción de validez (porque todavía está bien formado).

3.1 PI, comentarios y elementos externos al documento

Documento de entrada	<?xml version="1.0"?> <?xml-stylesheet href="doc.xsl" type="text/xsl" ?> <!DOCTYPE doc SYSTEM "doc.dtd"> <doc>Hello, world!<!-- Comment 1 --></doc> <?pi-without-data ?> <!-- Comment 2 --> <!-- Comment 3 -->
Forma canónica (sin comentar)	<?xml-stylesheet href="doc.xsl" type="text/xsl" ?> <doc>Hello, world!</doc> <?pi-without-data?>
Forma canónica (comentada)	<?xml-stylesheet href="doc.xsl" type="text/xsl" ?> <doc>Hello, world!<!-- Comment 1 --></doc> <?pi-without-data?> <!-- Comment 2 --> <!-- Comment 3 -->

Demuestra:

- Pérdida de declaración XML
- Pérdida de DTD
- Normalización de espacios en blanco fuera del elemento del documento (el primer carácter de ambas formas canónicas es '<'; los saltos de línea separan los PI y los comentarios fuera del elemento del documento)
- Pérdida de espacios en blanco entre PItarget y sus datos.
- Retención de espacios en blanco dentro de los datos de PI
- Eliminación de comentarios de la forma canónica sin comentarios, incluido el delimitador de comentarios fuera del elemento del documento (el último carácter

en ambas formas canónicas es '>')

3.2 Espacios en blanco en el contenido del documento

Documento de entrada

```
<doc>
   <clean>   </clean>
   <dirty>   A   B   </dirty>
   <mixed>
      A
      <clean>   </clean>
      B
      <dirty>   A   B   </dirty>
      C
   </mixed>
</doc>
```

Forma canónica

```
<doc>
   <clean>   </clean>
   <dirty>   A   B   </dirty>
   <mixed>
      A
      <clean>   </clean>
      B
      <dirty>   A   B   </dirty>
      C
   </mixed>
</doc>
```

Demuestra:

- Conserve todos los espacios en blanco entre etiquetas de inicio consecutivas, limpias o sucias.
- Conserve todos los espacios en blanco entre etiquetas finales consecutivas, limpias o sucias.
- Conserve todos los espacios en blanco entre el par de etiqueta final/etiqueta inicial, limpios o sucios
- Conservar todos los espacios en blanco en el contenido de los caracteres, limpios o sucios.

Nota: En este ejemplo, el documento de entrada y la forma canónica son idénticos. Ambos terminan con el carácter '>'.

3.3 Etiquetas de inicio y fin

Documento de entrada

```
<!DOCTYPE doc [<!ATTLIST e9 attr CDATA "default">]>
<doc>
   <e1   />
   <e2   ></e2>
   <e3   name = "elem3"   id="elem3"   />
   <e4   name="elem4"   id="elem4"   ></e4>
   <e5 a:attr="out" b:attr="sorted" attr2="all"
attr="I'm"
      xmlns:b="http://www.ietf.org"
      xmlns:a="http://www.w3.org"
      xmlns="http://example.org"/>
   <e6 xmlns="" xmlns:a="http://www.w3.org">
      <e7 xmlns="http://www.ietf.org">
         <e8 xmlns="" xmlns:a="http://www.w3.org">
            <e9 xmlns=""
xmlns:a="http://www.ietf.org"/>
         </e8>
      </e7>
```

```
   </e6>
</doc>
```

Forma canónica

```
<doc>
   <e1></e1>
   <e2></e2>
   <e3 id="elem3" name="elem3"></e3>
   <e4 id="elem4" name="elem4"></e4>
   <e5 xmlns="http://example.org"
xmlns:a="http://www.w3.org"
xmlns:b="http://www.ietf.org" attr="I'm" attr2="all"
b:attr="sorted" a:attr="out"></e5>
   <e6 xmlns:a="http://www.w3.org">
      <e7 xmlns="http://www.ietf.org">
         <e8 xmlns="">
            <e9 xmlns:a="http://www.ietf.org"
attr="default"></e9>
         </e8>
      </e7>
   </e6>
</doc>
```

Demuestra:

- Conversión de elementos vacíos a par de etiquetas inicio-fin
- Normalización de espacios en blanco en etiquetas de inicio y fin.
- Orden relativo del espacio de nombres y los ejes de atributos.
- Orden lexicográfico de espacios de nombres y ejes de atributos.
- Conservación de prefijos de espacios de nombres del documento original
- Eliminación de declaraciones de espacios de nombres superfluas
- Adición de atributo predeterminado

Nota: Algunas etiquetas de inicio en la forma canónica son muy largas, pero cada etiqueta de inicio en este ejemplo está enteramente en una sola línea.

Nota: En `e5`, `b:attr`precede `a:attr`porque la clave principal es el URI del espacio de nombres, no el prefijo del espacio de nombres, y `attr2`precede `b:attr`porque el espacio de nombres predeterminado no se aplica a atributos no calificados (por lo que el URI del espacio de nombres `attr2`está vacío).

3.4 Modificaciones de personajes y referencias de personajes

Documento de entrada

```
<!DOCTYPE doc [
<!ATTLIST normId id ID #IMPLIED>
<!ATTLIST normNames attr NMTOKENS #IMPLIED>
]>
<doc>
   <text>First line&#x0d;&#10;Second line</text>
   <value>&#x32;</value>
   <compute><![CDATA[value>"0" && value<"10"
?"valid":"error"]]></compute>
   <compute expr='value>"0" &amp;&amp; value&lt;"10"
?"valid":"error"'>valid</compute>
   <norm attr=' &apos;   &#x20;&#13;&#xa;&#9;   &apos;
'/>
   <normNames
attr='   A   &#x20;&#13;&#xa;&#9;   B   '/>
   <normId id=' &apos;   &#x20;&#13;&#xa;&#9;   &apos;
'/>
</doc>
```

Forma canónica

```
<doc>
   <text>First line&#xD;
```

```
Second line</text>
   <value>2</value>
   <compute>value&gt;"0" &amp;&amp; value&lt;"10"
?"valid":"error"</compute>
   <compute expr="value>&quot;0&quot; &amp;&amp;
value&lt;&quot;10&quot; ?
&quot;valid&quot;:&quot;error&quot;">valid</compute>
   <norm attr=" '    &#xD;&#xA;&#x9;    ' "></norm>
   <normNames attr="A &#xD;&#xA;&#x9; B"></normNames>
   <normId id="' &#xD;&#xA;&#x9; '"></normId>
</doc>
```

Demuestra:

- Reemplazo de referencia de personaje
- Delimitadores de valor de atributo establecidos entre comillas (comillas dobles)
- Normalización del valor del atributo
- Reemplazo de sección CDATA
- Codificación de caracteres especiales como referencias de caracteres en valores de atributos (&, <, ", , ,)
- Codificación de caracteres especiales como referencias de caracteres en texto (&, <, >,)

Nota: El último elemento, `normId`está bien formado pero viola una restricción de validez para los atributos de tipo ID. Para probar implementaciones XML canónicas basadas en procesadores de validación, elimine la línea que contiene este elemento de la forma canónica y de entrada. En general, se debe disuadir a los consumidores de XML de utilizar esta característica de XML.

Nota: Las referencias de caracteres de espacios en blanco que no sean no se ven afectados por la normalización del valor del atributo [XML] .

Nota: En la forma canónica, el valor del atributo nombrado `attr`en el elemento `norm`comienza con un espacio, un apóstrofo (comilla simple) y luego *cuatro* espacios antes de la referencia del primer carácter.

Nota: El `expr`atributo del segundo `compute`elemento no contiene saltos de línea.

3.5 Referencias de entidades

Documento de entrada

```
<!DOCTYPE doc [
<!ATTLIST doc attrExtEnt ENTITY #IMPLIED>
<!ENTITY ent1 "Hello">
<!ENTITY ent2 SYSTEM "world.txt">
<!ENTITY entExt SYSTEM "earth.gif" NDATA gif>
<!NOTATION gif SYSTEM "viewgif.exe">
]>
<doc attrExtEnt="entExt">
   &ent1;, &ent2;!
</doc>

<!-- Let world.txt contain "world" (excluding the
quotes) -->
```

Forma canónica (sin comentar)

```
<doc attrExtEnt="entExt">
   Hello, world!
</doc>
```

Demuestra:

- Reemplazo de referencia de entidad analizada interna
- Reemplazo de referencia de entidad analizada externa (incluidos espacios en blanco fuera de elementos y PI)
- Referencia de entidad externa no analizada

3.6 Codificación UTF-8

<table>
<tr><td>Documento de entrada</td><td><pre><?xml version="1.0" encoding="ISO-8859-1"?>
<doc>©</doc></pre></td></tr>
<tr><td>Forma canónica</td><td><pre><doc>#xC2#xA9</doc></pre></td></tr>
</table>

Demuestra:

- Efecto de la transcodificación de una codificación de muestra a UTF-8

Nota: El contenido del elemento doc NO es la cadena #xC2#xA9 sino los dos octetos cuyos valores hexadecimales son C2 y A9, que es la codificación UTF-8 del punto de código UCS para el signo de copyright (©).

3.7 Subconjuntos de documentos

<table>
<tr><td>Documento de entrada</td><td><pre><!DOCTYPE doc [
<!ATTLIST e2 xml:space (default|preserve) 'preserve'>
<!ATTLIST e3 id ID #IMPLIED>
]>
<doc xmlns="http://www.ietf.org"
xmlns:w3c="http://www.w3.org">
   <e1>
      <e2 xmlns="">
         <e3 id="E3"/>
      </e2>
   </e1>
</doc></pre></td></tr>
<tr><td>Expresión de subconjunto de documentos</td><td><pre><!-- Evaluate with declaration
xmlns:ietf="http://www.ietf.org" -->

(//. | //@* | //namespace::*)
[
   self::ietf:e1 or (parent::ietf:e1 and
not(self::text() or self::e2))
   or
   count(id("E3")|ancestor-or-self::node()) =
count(ancestor-or-self::node())
]</pre></td></tr>
<tr><td>Forma canónica</td><td><pre><e1 xmlns="http://www.ietf.org"
xmlns:w3c="http://www.w3.org"><e3 xmlns="" id="E3"
xml:space="preserve"></e3></e1></pre></td></tr>
</table>

Demuestra:

- Propagación del espacio de nombres predeterminado vacío desde el elemento principal omitido
- Propagación de atributos en el `xml`espacio de nombres en subconjuntos de documentos.
- Persistencia de declaraciones de espacios de nombres omitidas en descendientes

Nota: En la expresión del subconjunto del documento, la subexpresión `(//. | //@* | //namespace::*)`selecciona todos los nodos en el documento de entrada, sometiendo cada uno a la expresión de predicado entre corchetes. La expresión es verdadera para `e1`y sus nodos de espacio de nombres implícitos, y es verdadera si el elemento identificado por E3 está en la `ancestor-or-self` ruta del nodo de contexto (de modo que ancestro o yo permanezca del mismo tamaño bajo la unión con el elemento identificado por E3) .

Nota: La forma canónica no contiene delimitadores de línea.

4 resoluciones

Esta sección analiza una serie de puntos de decisión clave, así como una justificación para cada decisión. Aunque esta especificación ahora define la canonicalización XML en términos del modelo de datos XPath en lugar de XML Infoset , la forma canónica descrita en este documento es bastante similar en la mayoría de los aspectos a la forma canónica descrita en el borrador XML canónico de enero de 2000 [C14N-20000119] . Sin embargo, existen algunas diferencias y varias de las subsecciones analizan los cambios.

4.1 Sin declaración XML

La declaración XML, incluido el número de versión y la codificación de caracteres, se omite en la forma canónica. La codificación no es necesaria ya que la forma canónica está codificada en UTF-8. La versión no es necesaria ya que la ausencia de un número de versión indica inequívocamente XML 1.0.

Las versiones futuras de XML deberán incluir una declaración XML para indicar el número de versión. Sin embargo, es posible que el método de canonicalización descrito en esta especificación no sea aplicable a versiones futuras de XML sin algunas modificaciones. Cuando se requiere la canonicalización de una nueva versión de XML, esta especificación podría actualizarse para incluir la declaración XML, ya que presumiblemente la ausencia de la declaración XML del modelo de datos XPath se puede remediar en ese momento (por ejemplo, reemitiendo un nuevo XPath basado en el modelo de datos de conjunto de información).

4.2 Normalización del modelo sin caracteres

El estándar Unicode [Unicode] permite múltiples representaciones diferentes de ciertos "caracteres precompuestos" (un ejemplo simple es "ç"). Por lo tanto, dos documentos XML con contenido equivalente para la mayoría de las aplicaciones pueden contener secuencias de caracteres diferentes. El W3C está preparando una representación normalizada [CharModel] . El borrador XML canónico C14N-20000119 utilizó esta forma normalizada. Sin embargo, muchos procesadores XML 1.0 no realizan esta normalización. Además, las aplicaciones que deben resolver este problema normalmente imponen la normalización del modelo de caracteres en todo momento, comenzando cuando se crea el contenido del carácter, para evitar fallas de procesamiento que de otro modo podrían resultar (por ejemplo, consulte el ejemplo de Cowan) . Por lo tanto, la normalización del modelo de caracteres ha quedado fuera del alcance de la canonicalización XML. Sin embargo, el procesador XML utilizado para preparar la entrada del modelo de datos XPath debe (según el modelo de datos) utilizar el formulario de normalización C [NFC , NFC-Corrigendum] al convertir un documento XML al dominio de caracteres UCS desde cualquier codificación que no sea UCS. -basado (actualmente, las codificaciones basadas en UCS incluyen UTF-8, UTF-16, UTF-16BE y UTF-16LE, UCS-2 y UCS-4).

4.3 Manejo de espacios en blanco fuera del elemento del documento

El borrador XML canónico C14N-20000119 colocó un #xA después de cada PI fuera del elemento del documento, así como un #xA después de la etiqueta final del elemento del documento. El método en esta especificación realiza la misma función excepto por omitir el #xA final después del último PI (o comentario o etiqueta final del elemento del documento). Esta técnica garantiza que los elementos secundarios de PI (y comentarios) de la raíz estén separados del marcado mediante un avance de línea incluso si el nodo raíz o el elemento del documento se omiten del conjunto de nodos de salida.

4.4 Sin reescritura de prefijo de espacio de nombres

El borrador XML canónico C14N-20000119 describía un método para reescribir prefijos de espacios de nombres de modo que dos documentos que tuvieran declaraciones de espacios de nombres lógicamente equivalentes también tuvieran prefijos de espacios de nombres idénticos. El objetivo era eliminar la dependencia de los prefijos de espacios de nombres particulares en un documento al realizar pruebas de equivalencia lógica. Sin embargo, ahora existen varios contextos en los que los prefijos de espacios de nombres pueden impartir valor informativo en un documento XML. Por ejemplo, una expresión XPath en el valor de un atributo o en el contenido de un elemento puede hacer referencia a un prefijo de espacio de nombres. Por lo tanto, reescribir los prefijos del espacio de nombres dañaría dicho documento al cambiar su significado (y no puede ser lógicamente equivalente si su significado ha cambiado).

Más formalmente, supongamos que D1 sea un documento que contiene un XPath en un valor de atributo o contenido de elemento que hace referencia a los prefijos de espacios de nombres utilizados en D1. Supongamos además que todos los prefijos de espacio de nombres en D1 se reescribirán mediante el método de canonicalización. Sea D2 = D1, luego modifique los prefijos de espacio de nombres en D2 y modifique las referencias de la expresión XPath a los prefijos de espacio de nombres de modo que D2 y D1 sigan siendo lógicamente equivalentes. Dado que la reescritura de espacios de nombres no incluye apariciones de referencias de espacios de nombres en valores de atributos y contenido de elementos, la forma canónica de D1 no es igual a la forma canónica de D2 porque el XPath será diferente. Por lo tanto, aunque la reescritura de espacios de nombres normaliza las declaraciones de espacios de nombres, no se logra el objetivo de eliminar la dependencia de los prefijos de espacios de nombres particulares en el documento.

Además, es posible demostrar que la reescritura de espacios de nombres es perjudicial, en lugar de simplemente ineficaz. Sea D1 un documento que contiene un XPath en un valor de atributo o contenido de elemento que hace referencia a los prefijos de espacio de nombres utilizados en D1. Supongamos además que todos los prefijos de espacio de nombres en D1 se reescribirán mediante el método de canonicalización. Ahora sea D2 la forma canónica de D1. Claramente, las formas canónicas de D1 y D2 son equivalentes (ya que D2 es la forma canónica de la forma canónica de D1), sin embargo, D1 y D2 no son lógicamente equivalentes porque el XPath antes mencionado funciona en D1 y no en D2.

Tenga en cuenta que se puede presentar un argumento similar a este contra el método de canonicalización XML basado en cualquiera de los casos de las Limitaciones ; los problemas no se pueden solucionar fácilmente en esos casos, mientras que aquí tenemos la oportunidad de evitar introducir intencionadamente dicha limitación.

Las aplicaciones que deben probar la equivalencia lógica deben realizar pruebas más sofisticadas que la mera comparación de flujos de octetos. Sin embargo, es muy probable que esto sea necesario en cualquier caso para probar equivalencias lógicas basadas en reglas de aplicación, así como reglas de otras recomendaciones, borradores de trabajo y trabajos futuros relacionados con XML.

4.5 Orden de las declaraciones y atributos del espacio de nombres

El borrador XML canónico C14N-20000119 alternaba entre declaraciones de espacios de nombres y declaraciones de atributos. Esto es parte del esquema de reescritura de prefijos de espacios de nombres, que esta especificación elimina. Esta especificación sigue el modelo de datos XPath de colocar todos los nodos del espacio de nombres antes que todos los nodos de atributos.

4.6 Declaraciones de espacios de nombres superfluos

Las declaraciones de espacios de nombres innecesarias no se realizan en forma canónica. Ya sea para un espacio de nombres predeterminado vacío, un espacio de nombres predeterminado no vacío o un enlace de prefijo de espacio de nombres, el método de canonicalización XML omite una declaración si determina que el elemento principal inmediato *en la forma canónica* tiene una declaración equivalente en alcance. El elemento del documento raíz se maneja de manera especial ya que no tiene ningún elemento principal. Todas las declaraciones de espacios de nombres que contiene se conservan, excepto la declaración de un espacio de nombres predeterminado vacío que se omite automáticamente.

En relación con el método de simplemente representar todo el contexto del espacio de nombres de cada elemento, las implementaciones no se ven obstaculizadas por más que un factor constante en el tiempo de procesamiento y el uso de la memoria. Las ventajas incluyen:

- Elimina la saturación de `xmlns=""`formas canónicas de aplicaciones que pueden ni siquiera utilizar espacios de nombres o admitirlos sólo mínimamente.
- Elimina las declaraciones de espacios de nombres de elementos a los que pueden no pertenecer según el modelo de contenido de la aplicación, simplificando así la tarea de volver a adjuntar una declaración de tipo de documento a un formulario canónico.

Tenga en cuenta que en los subconjuntos de documentos, un elemento con omisiones en su cadena de elementos ancestrales se representará en la forma canónica con declaraciones de espacio de nombres que pueden haberse realizado en sus ancestros omitidos, preservando así el significado del elemento.

4.7 Propagación de la declaración de espacio de nombres predeterminada en subconjuntos de documentos

El modelo de datos XPath representa un espacio de nombres predeterminado vacío con la ausencia de un nodo, no con la presencia de un nodo de espacio de nombres predeterminado que tenga un valor vacío. Por lo tanto, con respecto al hecho de que el elemento `e3`en los siguientes ejemplos no está calificado como espacio de nombres, no podemos distinguir entre `<e1 xmlns="a:b"><e2 xmlns=""><e3/></e2></e1>` versus `<e1 xmlns="a:b"><e2><e3 xmlns=""/></e2></e1>`. Todo lo que sabemos es que `e3`no había un espacio de nombres calificado en la entrada, por lo que conservamos esta información

en la salida si `e2`se omite para que `e3` no adopte la calificación de espacio de nombres predeterminada de `e1`.

4.8 Ordenar atributos por URI de espacio de nombres

Dado el requisito de preservar los prefijos de espacio de nombres declarados en un documento, ordenar los atributos con el prefijo, en lugar del URI del espacio de nombres, como clave principal, es viable y más fácil de implementar. Sin embargo, se seleccionó el URI del espacio de nombres como clave principal porque se acerca más a la intención de la especificación Espacios de nombres en XML , que es identificar espacios de nombres por URI y nombre local, no por un prefijo y un nombre local. El efecto de la clasificación es agrupar todos los atributos que se encuentran en el mismo espacio de nombres.

5 referencias

C14N-20000119
: *Canonical XML Versión 1.0* , Borrador de trabajo del W3C. T. Bray, J. Clark, J. Tauber y J. Cowan. 19 de enero de 2000. http://www.w3.org/TR/2000/WD-xml-c14n-20000119.html .

Modelo de carbón
: *Modelo de caracteres para la World Wide Web* , Borrador de trabajo del W3C. editores. Martin J. Dürst, François Yergeau, Misha Wolf, Asmus Freytag y Tex Texin. http://www.w3.org/TR/charmod/ .

Cowan
: *Ejemplo de efecto nocivo de la normalización del modelo de caracteres* , carta en el archivo de correo del grupo de trabajo de firma XML. John Cowan, 7 de julio de 2000. http://lists.w3.org/Archives/Public/w3c-ietf-xmldsig/2000JulSep/0038.html .

Conjunto de información
: *Conjunto de información XML* , borrador de trabajo del W3C. editores. John Cowan y Richard Tobin. http://www.w3.org/TR/xml-infoset .

ISO-8859-1
: *ISO-8859-1 Juego de caracteres latino 1* . http://www.utoronto.ca/webdocs/HTMLdocs/NewHTML/iso_table.html o http://www.iso.ch/cate/cat.html .

Palabras clave
: *Palabras clave para su uso en RFC para indicar niveles de requisitos* , IETF RFC 2119. S. Bradner. Marzo de 1997. http://www.ietf.org/rfc/rfc2119.txt .

Espacios de nombres
: *Espacios de nombres en XML* , Recomendación W3C. editores. Tim Bray, Dave Hollander y Andrew Layman. http://www.w3.org/TR/REC-xml-names/ .

NFC
: *TR15, Formularios de normalización Unicode.* M. Davis, M. Durst. Revisión 18: noviembre de 1999. http://www.unicode.org/unicode/reports/tr15/tr15-18.html .

Corrección NFC
: *Corrigendum de normalización* . El Consorcio Unicode. http://www.unicode.org/unicode/uni2errata/Normalization_Corrigendum.html .

Unicódigo
: *El estándar Unicode, versión 3.0.* El Consorcio Unicode. ISBN 0-201-61633-5. http://www.unicode.org/unicode/standard/versions/Unicode3.0.html .

UTF-16
: *UTF-16, una codificación de ISO 10646* , IETF RFC 2781. P. Hoffman, F. Yergeau. Febrero de 2000. http://www.ietf.org/rfc/rfc2781.txt .

UTF-8

UTF-8, un formato de transformación de ISO 10646 , IETF RFC 2279. F. Yergeau. Enero de 1998. http://www.ietf.org/rfc/rfc2279.txt .

URI

Identificadores uniformes de recursos (URI): sintaxis genérica , IETF RFC 2396. T. Berners-Lee, R. Fielding, L. Masinter. Agosto de 1998 http://www.ietf.org/rfc/rfc2396.txt .

Xbase

Edición base XML . Jonathan Marsh. 7 de junio de 2000. http://www.w3.org/TR/xmlbase/ .

XML

Lenguaje de marcado extensible (XML) 1.0 (segunda edición) , recomendación del W3C. editores. Tim Bray, Jean Paoli, CM Sperberg-McQueen y Eve Maler. 6 de octubre de 2000. http://www.w3.org/TR/REC-xml .

XML DSig

Sintaxis y procesamiento de firmas XML , borrador del IETF/recomendación candidata del W3C. D. Eastlake, J. Reagle, D. Solo, M. Bartel, J. Boyer, B. Fox y E. Simon. 31 de octubre de 2000. http://www.w3.org/TR/xmldsig-core/ .

Decisión Plenaria XML

Decisión plenaria XML del W3C sobre referencias de URI relativas en declaraciones de espacios de nombres , documento del W3C. 11 de septiembre de 2000. http://lists.w3.org/Archives/Public/xml-uri/2000Sep/0083.html .

XPath

Lenguaje de ruta XML (XPath) versión 1.0 , recomendación W3C. editores. James Clark y Steven DeRose. 16 de noviembre de 1999. http://www.w3.org/TR/1999/REC-xpath-19991116 .

6 Agradecimientos (Informativo)

Las siguientes personas brindaron comentarios valiosos que mejoraron la calidad de esta especificación:

- Doug Bunting, Ariba
- John Cowan, Reuters
- Martín J. Durst, W3C
- Donald Eastlake 3.º, Motorola
- Merlín Hughes, Baltimore
- Gregor Karlinger, IAIK TU Graz
- Susan Lesch, W3C
- Jonathan Marsh, Microsoft
- José Reagle, W3C
- Petteri Stenius, Hecho360
- Kent TAMURA, IBM